BAB 3

METODOLOGI PENELITIAN

3.1 Metodologi Penelitian

Penelitian "Rancang Bangun Browser Extension untuk Website Phising Detector Menggunakan Algoritma XGBoost" menggunakan beberapa tahap agar metodologi dan perancangan sistem dapat terpenuhi. Tahap-tahap yang dilaksanakan antara lain adalah sebagai berikut.

a. Identifikasi dan Perumusan Masalah

Permasalahan yang dipilih untuk diteliti adalah merancang dan membangun *browser extension* untuk *website phising detector* menggunakan algoritma XGBoost dan nilai *precision, recall*, dan F1 *score* pada pendeteksian *website phising.*

b. Studi Literatur

Berdasarkan penelitian yang dipilih, selanjutnya dilakukan studi literatur yang bertujuan untuk memperoleh informasi dan teori-teori yang berhubungan dengan permasalahan tersebut, seperti jurnal dari penelitian mengenai topik terkait dan serupa yang sudah pernah dilakukan sebelumnya. Proses ini bertujuan untuk meningkatkan pemahaman secara teori terkait algoritma XGBoost dan *website phising* untuk memenuhi landasan teori penelitian.

c. Pengumpulan Data

Dataset yang digunakan dalam penelitian diperoleh dari UCI Repository

Phising Websites dataset. Dataset berisi 11055 data dengan 31 atribut. Data yang ada sudah diberi label -1 untuk *website* yang bukan phising, 0 untuk *website* yang mencurigakan dan 1 untuk *website* yang phising.

d. Pemrograman Sistem

Berdasarkan topik permasalahan yang sudah dipilih, selanjutnya dilakukan analisis untuk mengimplementasikan algoritma yang tepat dan sesuai dengan permasalahan tersebut. Algoritma XGBoost digunakan dalam penelitian ini. Kemudian dibangun sebuah *script* python yang digunakan untuk melakukan ekstraksi fitur dari *url*. XGBoost digunakan untuk melakukan klasifikasi terhadap hasil ekstraksi fitur yang sudah diperoleh.

e. Implementasi Algoritma

Pada tahap ini, XGBoost diimplementasikan untuk melakukan identifikasi *website phising* dengan menggunakan *dataset* dari UCI. Untuk mendapatkan nilai parameter terbaik, digunakan pemilihan fitur menggunakan *Recursive Feature Elimination* (RFE) dan optimasi *hyperparameter* menggunakan *Random Search.*

f. Evaluasi Hasil Implementasi Algoritma

Setelah algoritma diimplementasikan, hal selanjutnya dilakukan adalah melakukan evaluasi terhadap akurasi yang dihasilkan dari implementasi algoritma XGBoost dalam melakukan identifikasi *website phising*. Adapun evaluasi dilakukan dengan menggunakan F1 *score* di mana kemampuan dari model yang dihasilkan untuk melakukan klasifikasi dinilai berdasarkan *precission* dan *recall* dari model. Skor F1 merupakan bilangan kontinu dengan rentang dari 0 sampai

dengan 1. Apabila dihasilkan F1 *score* yang baik, maka hal tersebut mengindikasikan bahwa model klasifikasi yang dibuat memiliki *precision* dan *recall* yang baik.

g. Penyusunan Laporan

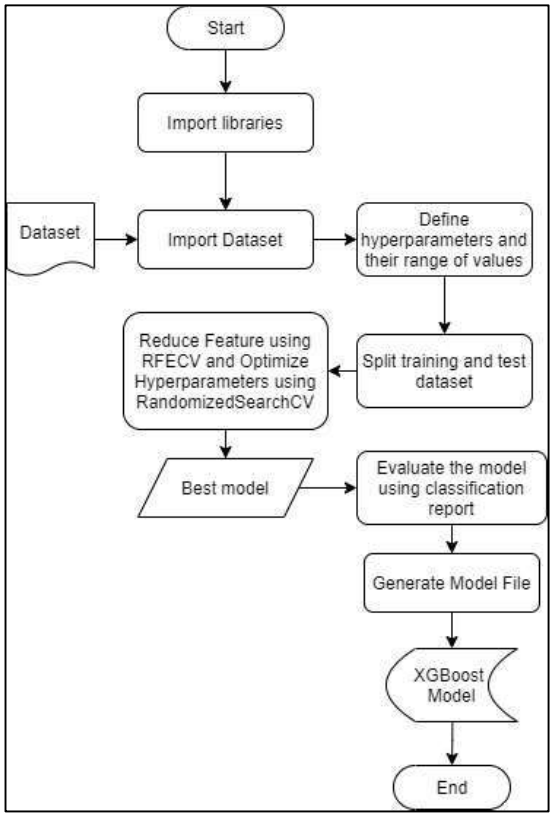
Tahap terakhir dalam penelitian ini adalah menyusun laporan yang berisikan hasil keseluruhan dari penelitian yang sudah dilakukan sebagai dokumentasi dari penelitian yang sudah dilakukan. Laporan disusun secara terstruktur sesuai dengan kaidah penyusunan dan penulisan laporan ilmiah yang sudah ditentukan, dimulai dari pendahuluan hingga kesimpulan dan saran.

3.2 Perancangan Sistem

Sistem yang dihasilkan dari penelitian ini berbasis ekstensi *browser* dengan menggunakan HTML dan *JavaScript*, beserta sebuah *web service* berbasis *flask*. Sebelum ekstensi *browser* dan *web service* dibuat, dilakukan beberapa perancangan beberapa *flowchart* yang terdiri dari *flowchart* untuk pembuatan model XGBoost, *flowchart* program ekstensi *browser*, *flowchart* ekstraksi dan prediksi pada *web service*, dan rancangan tampilan antarmuka ekstensi browser.

3.2.1 Flowchart Pembuatan File Model XGBoost

Untuk dapat mengaplikasikan model yang sudah dibuat, model yang sudah didapatkan parameter terbaiknya diekspor menjadi sebuah file model berekstensi pkl menggunakan modul *python* bernama *pickle*.



Gambar 3.1 Gambaran Alur Pembuatan Model XGBoost

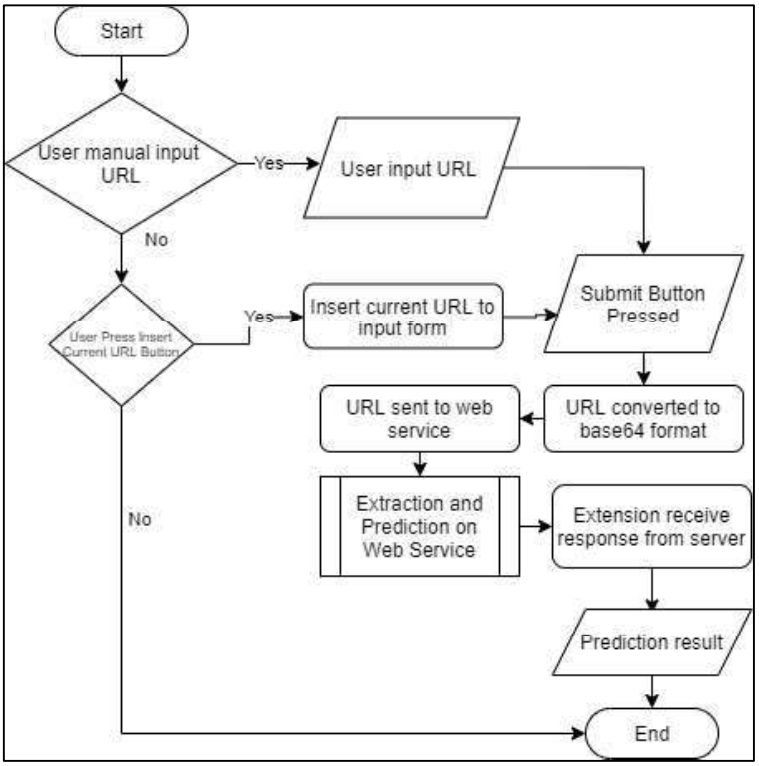
Pertama, *library* yang dibutuhkan di*import*, kemudian dilanjutkan dengan *inport dataset* yang sudah ada. Setelah selesai, kemudian melakukan pendefinisian pada *value* dari *hyperparameter*. Setelah menentukan *hyperparameter* dan rentang nilai untuk setiap *hyperparameter* tersebut, *dataset* yang digunakan pada penelitian ini dibagi menjadi *dataset train* dan *test.* Setelah itu dilakukan *feature selection* menggunakan *Recursive Feature Elimination*. Pada *feature selection* digunakan strategi *cross validation Stratified KFold* dengan *fold* sebesar tiga (3). Pembagian *dataset* menggunakan *cross validation* pada umumnya menghasilkan model yang lebih tidak bias apabila dengan metode lain, seperti misalnya penggunaan train_test_split sederhana. Untuk melakukan optimasi terhadap nilai dari

hyperparameter digunakan *RandomizedSearchCV*. Pada *RandomizedSearchCV* digunakan *cross validation Stratified KFold* dengan *fold* sebesar tiga (3) dan jumlah iterasi yang dilakukan sebanyak seratus (100).

Setelah dilakukan optimasi *hyperparameter* dan dihasilkan model terbaik, dilakukan evaluasi dengan menggunakan F1 *score* yang merupakan perbandingan rata - rata *precission* dan *recall* dari model. Tingkat sensitifitas atau *recall* dari model ditentukan oleh nilai dari *True Positive Rate* (TPR) yang diperoleh dari hasil perbandingan antara jumlah website yang berhasil diidentifikasian secara benar terindikasi *phising* dengan jumlah semua *website* yang memang terindikasi sebagai *website phising*. Sementara itu, tingkat spesifisitas atau *precission* ditentukan berdasarkan nilai dari *False Positive Rate* (FPR) yang diperoleh dari hasil perbandingan antara jumlah *website* yang diidentifikasikan bukan *website phising* dengan jumlah semua *website* yang memang terindikasi bukan *website phising*.

3.2.2 Flowchart Extensi Browser

Pengecekan *url* dimulai dengan mengirim sebuah *GET request* ke *web service checker* dengan sebuah *query* yaitu *url* yang berisi *url* yang berformat *base64*. Kemudian, *web service* menerima *request* tersebut dan mengecek *url* tersebut. Setelah selesai dicek, *web service* kemudian memberikan *response* ke *server* dalam bentuk *json* yang isinya berupa *integer*, dimana nilai -1 berarti bukan *website phising* dan nilai 1 untuk *website phising*. Adapun *flowchart* untuk proses pengecekan *website* dapat dilihat pada Gambar 3.2.

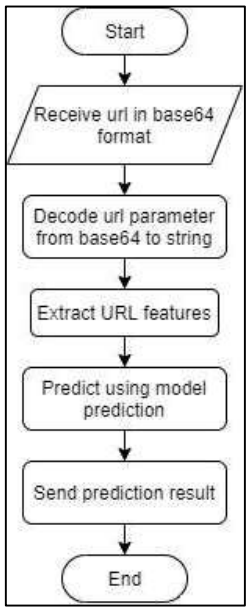


Gambar 3.2 *Flowchart* proses pengecekan *website* menggunakan ekstensi *browser*

Berdasarkan Gambar 3.2, *user* dapat memilih 2 metode input dari *url* yang diperiksa, yaitu melalui *input* secara manual atau memeriksa *website* yang sedang dibuka sekarang. Apabila *user* memilih untuk memeriksa *website* yang sedang dibuka sekarang, ekstensi otomatis mengambil *url* pada tab yang sedang dibuka dan langsung mengirim *url* tersebut dalam bentuk *base64*. Kemudan setelah *web service* menerima *request* yang dikirim oleh ekstensi, dilanjutkan dengan proses pemeriksaan *url.* Setelah selesai diperiksa, *web service* mengirim kembali hasil dari prediksinya.

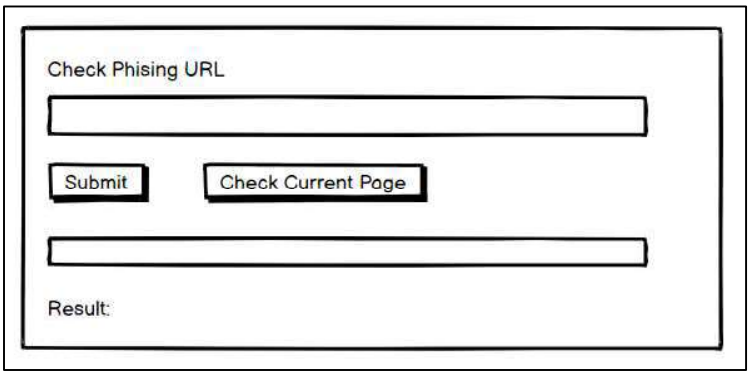
3.2.3 Flowchart Proses Ekstraksi dan Prediksi pada *Web Service*

Pada saat *web service* menerima request dari *browser*, *query* dari *url* diambil kemudian dilakukan *decode* dari base64 menjadi *string url*. Proses ekstraksi kemudian dijalankan dan fitur dari *website* diekstrak. Setelah proses ekstraksi selesai, *data* hasil ekstraksi kemudian dijadikan sebagai nilai untuk dimasukkan ke dalam model prediksi. Model prediksi kemudian mengeluarkan hasil prediksi berdasarkan nilai yang dimasukkan berupa -1 untuk *website* yang bukan *phising* dan nilai 1 untuk *website phising.* Dari hasil nilai prediksi ini selanjutnya dikirimkan kembali ke ekstensi *browser* untuk selanjutnya ditampilkan ke *user*. Proses ekstraksi dan prediksi dapat dilihat pada gambar 3.3.



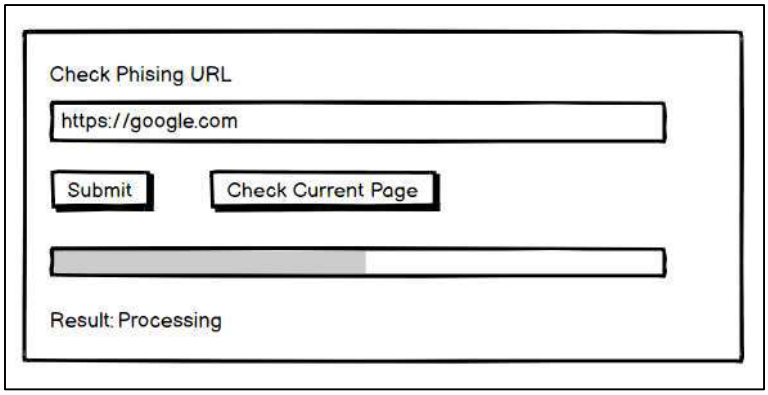
Gambar 3.3 *Flowchart extraction and prediction on web service*

3.2.4 Rancangan Tampilan Antarmuka Ekstensi Browser

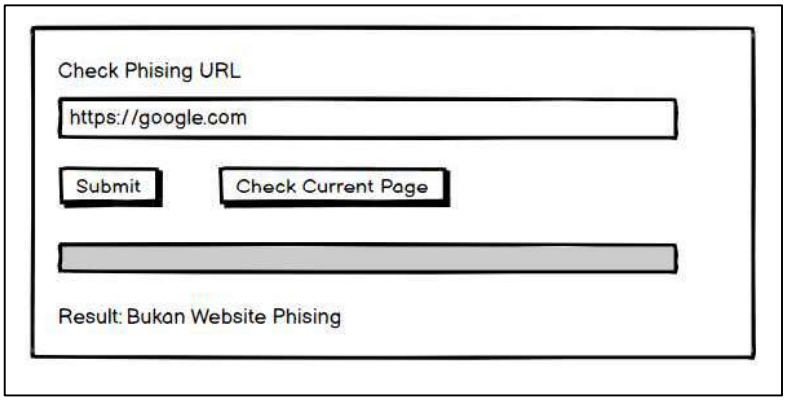


Gambar 3.4 Tampilan ekstensi *browser*

Tampilan antarmuka atau *interface* ekstensi browser terdapat sebuah form input untuk mengisi *url* yang diperiksa, terdapat juga sebuah tombol bertuliskan "Submit" untuk mengirim *url* ke *web service,* sebuah tombol bertuliskan "Check Current Page" untuk mengirim *url* dari tab yang sedang dibuka ke *web service*, sebuah progress bar untuk menunjukan progress dari pengecekan, dan bagian *result* yang menunjukan hasil dari prediksi.



Gambar 3.5 Tampilan ekstensi *browser* saat sedang memeriksa *url* untuk diprediksi



Gambar 3.6 Tampilan ekstensi *browser* setelah menerima hasil prediksi